Unit3 training guidebook

1. 训练目的

- 掌握数据抽象的概念及要点，根据需求撰写规格以及代码
- 掌握JUnit单元测试的框架，利用JUnit开展单元测试

2. 知识要点

2.1 数据抽象

类的不变式（Invariant）定义了对象有效的判定条件，表现为类成员变量需要满足的约束。类的任何操作都不能导致不变式为假，否则就破坏了对象的有效性。在一个无效对象上的任何操作都不能保证获得正确的结果。不论一个类管理着什么具体数据，从设计角度来看，都具有一个不变式。否则，这个类的有效性就是未定义的，导致无法确定程序在运行时的正确性。例如，对于一个银行账户集合，其中一个类不变量为所有交易的总量，应该和余额总计相等。这样的不变量的约束通常在程序生命周期中一直生效。

为了在程序运行时针对对象有效性进行检查，可以把不变式中的约束条件用方法来实现（例如，`repOK()`），该方法不带参数，返回布尔值。测试时该类的每一个方法都应该使用它来确认对象的有效性。一般推荐在进入方法和退出时都加以调用。如果该方法执行之前，得到返回值false，说明对象无效，则可直接退出，并报告调用者；如果在方法正常退出之前的调用返回false，则表明当前方法本身的实现有缺陷，导致对象状态无效。

2.2 基于规格的测试用例设计

面向对象程序的基本构造单位是类，类测试是面向对象软件测试的关键。类测试与传统的单元测试相似，但由于类包含了属性和操作，实例化的对象具有状态特征，因此，和传统的单元测试不一样的是，必须要考虑对象的状态来设计测试。这使得类测试不能孤立地测试单个操作，而要将操作作为类的一部分，并同时结合对象的状态及状态之间的迁移，来设计对方法的调用和所需配置的相关参数数据。此外，类之间具有协作交互关系，在测试中往往需要构造多个对象来测试它们之间的协同。因此，本次实验训练根据类规格和方法规格来进行测试设计，要求分析规格说明，识别对象的状态，并针对方法的前置条件和后置条件来设计具体的测试用例。因此，规格不仅对于程序设计和实现具有重要意义，同样也是测试活动的重要依据。

在软件系统开发过程中，代码会频繁发生变化，会不断引入新的bug，这对每个开发人员的工作都会产生影响。为了确保整个系统的代码质量，在类和方法层次实施自动化测试不仅能够及时自动发现bug，还能够确保每个开发人员都在编写代码时及时更新或补充相应的测试代码，形成良性的质量保证思维。开源的java单元测试框架JUnit能大大简化进行单元测试所要做的工作。测试用例的设计分两种情况：（1）根据给定的方法规格来设计和编写针对单个方法的前置条件和后置条件的测试用例（例如`preOK()`和`afterOK()`）；（2）根据类的不变式，围绕对象状态设计综合性的测试用例。

3. 训练任务

以下规格说明描述了`IntSet`的数据抽象。`IntSet`对象可以用来存储和管理规模未知的一组无重复整数，每次修改只能操作不多于一个元素，且保持对象中存储的整数有序。其中：

- 方法`elementSwap()`完成两个`IntSet`对象所管理的数据集合的交换
- 方法`symmetricDifference()`完成求两个`IntSet`对象所管理的数据集合的对称差

具体的需求描述请见程序注释。

任务一：

补全 `elementSwap()` 及 `symmetricDifference()` 两个方法的规格，并填写代码。

任务二：

根据不变式在MySet.java文件中补全 `repOk()` 这一方法，并根据 `delete()` 方法和 `insert()` 方法的规格分析该方法的前置条件和后置条件，结合 `MySetTest` 类中的代码，填写 `deletepreOK()` 和 `insertafterOK()` 两个方法，使 `MySetTest` 类可以正确完全测试工作。要求：

- 对于 `deletepreOK()` 方法，如果当前删除的数不满足前置条件，需要输出 `Fail before we delete` + 此时删除的数字，例如 ”Fail before we delete 2“。
- 对于 `insertafterOK()` 方法，如果当前插入的数不满足后置条件，需要输出一行 `Fail after we insert` + 此时插入的数字，例如 ”Fail after we insert 0“。

任务三：

对已实现的函数 `insert()` 及 `delete()` 开展JUnit单元测试（运行 `TestRunner` 类），说明所找出bug的现象，同时直接在MySet.java文件中修改原码以消除bug。

要求：在测试前输出"Test Start!"，在测试后输出"Test End!"。

任务四：（作为拓展，不要求提交）

仿照 `MySetTest` 类中对于 `delete()` 和 `insert()` 两个方法的测试模式，设计对于任务一种填充完成的方法 `elementSwap()` 和 `symmetricDifference()` 的测试代码，并使用 JUnit 进行相应的测试。

所需提交的文件如下：

- IntSet.java：补全任务一中所要求规格以及代码的原IntSet.java文件。
- MySet.java：补全任务二中所要求方法，并修改完已实现函数中bug的原MySet.java文件。
- MySetTest.java：对MySet类开展JUnit单元测试的文件。
- README.md：对通过JUnit单元测试所找到的bug进行说明的文件。